



## *GeoFEST 4.5(G)*

### *Addendum to Design and User Documentation*

This document describes the functional and structural changes incorporated into the special version 4.5(G) release of GeoFEST. This version of the program was developed in order to demonstrate and study the use of solution-driven adaptive mesh refinement (using Pyramid) in large viscoelastic finite element problems. This version was used in March 2005 to study performance of a 16 million element mesh distributed over a cluster of 490 *Intel Pentium* processors. In addition to allowing interested users to duplicate and extend this work, posting of the source code for this version may provide a helpful starting point for researchers interested in development of similar but different mesh-modifying versions.

As was the case for the standard 4.5 release version, the code is targeted at hardware configurations of MPI-based parallel UNIX cluster computers. (Unlike the standard release however, this special purpose version does not offer the option of compiling for sequential computers; this release is for parallel use only.) Users should note that this code requires the latest version of Pyramid to properly compile and run (more recent than version 1.1.5).

The following summarizes changes in the execution flow and subroutine structure incorporated in this version:

The “`main.c`” module contains the top-level execution flow of the program. In the standard release, this consists of the “`input_phase`”, “`elastic`” and “`time_step`” tasks. In the current release, between the “`elastic`” and “`time_step`” tasks is inserted an automatic mesh refinement phase. This takes the form of the new tasks “`save_attributes`”, “`get_refine`” and “`update_phase`”. This mesh refinement step is followed by a repeat call to “`elastic`” which recomputes the elastic solution on the new mesh, after it has been refined based on the initial elastic solution. The subsequent viscoelastic solution proceeds as usual, through the “`time_step`” task on the updated mesh.

To support adaptive mesh refinement (AMR) for GeoFEST modifications were made to the files `finel.h`, `main.c`, and `solver.c` to support this feature for the elastic time step phase. The following represents changes to the listed files:

#### **finel.h:**

```
#define REFINE_THRESHOLD      1.99e-2
Strain energy threshold above which elements will be marked
for refinement.
```

## **main.c:**

### *Mesh Declaration*

An additional mesh variable was added to support both the initial mesh and the adaptively refined mesh as represented in the main program declaration section by:

```
pamr_Mesh[2];
```

### *Allocation of Storage for Mesh Components*

As node and element data must be stored for interpolation from the coarse mesh to the refined mesh, under load balancing and adaptive refinement, allocation for these variables has been assigned for both these meshes by the commands:

```
pamr_define_mesh_terms(&m[0], ...);  
pamr_define_mesh_terms(&m[1], ...);
```

### *Adaptive Refinement for Elastic Time Step*

Performing adaptive refinement requires that the coarse mesh node and element values (the current "solution") be saved for interpolation to the refined mesh. During the AMR stage a strain energy metric is applied where elements with values above a fixed threshold (defined by `REFINE_THRESHOLD` in `finel.h`) are marked for refinement. Subsequently, these coarse mesh elements are migrated in a load balanced way to new processors followed by the creation of new elements that represent the adaptive mesh. Now that two meshes exist, the coarse and the refined mesh, interpolation and definition of the AMR mesh can occur during an update phase to properly represent all aspects of the new mesh after refinement. These events occur during the following operations:

```
elastic(&m[0]) ;  
save_attributes(&m[0]) ;  
get_refine(&m[0], &m[1]) ;  
update_phase(&m[0], &m[1]) ;  
elastic(&m[1]) ;
```

Incidentally, once the `update_phase()` is completed the original coarse mesh is no longer needed so that storage is released.

## **solver.c:**

### *Communication Buffers in Globalize*

The solver works in parallel where shared node data must be

updated properly. Since numerous iterations are performed per time step on the mesh the routine "globalize()" only performs an initial local storage allocation reusing that memory over successive time steps. Since the mesh can now change due to AMR these coarse mesh buffers must be released and refined mesh buffers created and reused over successive time steps. Globalize now contains an additional variable that controls how these buffers are managed. These changes are integrated in the calls to globalize in the code, so no modifications are needed by the user.

The refinement is based on the strain energy of the elastic solution, or alternatively the formal elastic strain energy computed from the change in stress and strain from a single time step. This strategy is based on the insight that an optimal mesh for an elastic solution would have equal strain energy in every element in the mesh. We approximate this ideal by finding the elements with the most strain energy, and marking them for refinement. The resulting elements will each have lower strain energy than the original mesh, and so approach the ideal of strain energy equalization.

This strain energy is computed by a modification to the routine form\_stress. Strain energy is a quadratic form involving all the components of strain with the constitutive material tensor. We compute this by taking a dot-product of the stress with the strain. For time-stepping solutions, this is taken to be the single-step change in stress and strain, which is consistent with the view that a single step of the viscoelastic solution is analogous to an elastic solution with a modified right-hand side (loading).

New routines have been added to support the solution-based mesh refinement:

**Routine *save\_attributes*:**

Saves GeoFEST problem description values as PYRAMID attributes. This routine packages the parts of the GeoFEST mesh (BC's, split node definitions, material properties) into PYRAMID floating-point based records for propagation into the newly created mesh.

**Routine *get\_refine*:**

Performs adaptive mesh refinement based on strain energy. This routine marks elements for refinement based on the delta strain energy of the most recent solution (displacements, strains). Currently it uses a hard-wired threshold (REFINE\_THRESHOLD) which was designed to produce a 16-million element mesh from the original 10-million element mesh used in this case. The refined mesh is produced by invoking the pyramid routine pamr\_physical\_amr. Further processing is required to translate this refined mesh into active GeoFEST data structures (in update\_phase).

**Routine *update\_phase*:**

Retrieves attributes from PYRAMID refined mesh, and from that basis creates new GeoFEST internal mesh structures (nodes, elements, boundary conditions, split nodes, and material properties). Essentially every aspect of the mesh has changed since *input\_phase* read in the file-based mesh, so we need to refill all those data structures with valid data.